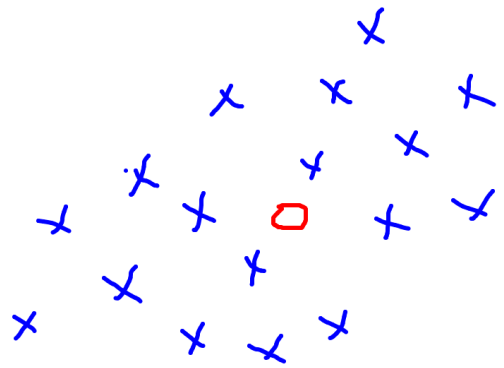
Lecture 8 Clustering

- Clustering
  - Quantization maximization
  - Expectation maximization

x =

1.0130	1.2629
2.7596	0.2501
1.3428	2.9105
1.3961	2.8671
2.1769	1.9201
1.6925	2.8985
1.8682	2.1837
2.5954	2.2908
3.0468	2.1129
1.8020	2.4400
2.3277	2.1017
1.7617	4.7873
2.2296	0.8333
2.4400	0.1457
1.3831	0.8593
2.2748	0.9067
2.6011	1.5664
2.0923	1.8315
3.7298	1.7815
1.3914	2.5413

Data generation
One center



```
>> x= [randn(20,2)+ones(20,1)*y(1,:)];
>> plot(x(:,1),x(:,2),'o')
>> centers=mean(x);
>> hold on;plot(centers(:,1),centers(:,2),'ro')
```

Forward data generation

$$y \in R^d$$

$$n_i = \text{randn}(1,2)$$

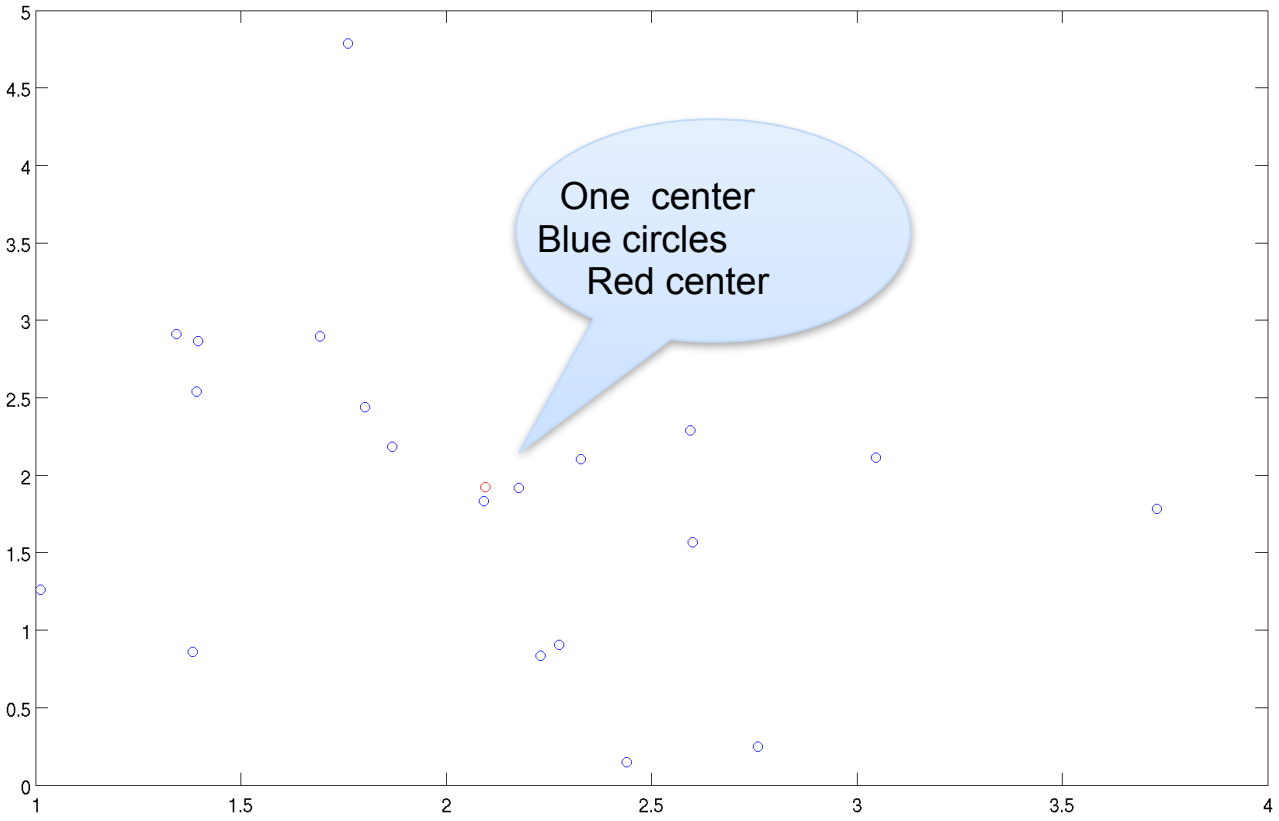
$$x_i = y + n_i$$

randn
A normal pdf
with zero mean
and
unit variance

Mean

$$S = \{ x_i \in R^d \}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$


One center
Blue circles
Red center
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
1.5
2
2.5
3
3.5
4

x =

2.7477	1.1392
1.7270	2.7847
3.5763	2.3086
1.5191	1.7661
2.3275	0.9430
2.6647	1.7159
2.0852	1.9133
2.8810	0.5306
2.3232	2.1922
1.2159	1.1777
0.1946	1.9058
3.8586	2.3362
1.3955	1.0953
2.1034	1.7117
2.5632	2.3501
2.1136	0.1641
1.0953	3.0360
1.5323	4.4245
1.8751	2.9594
3.4790	1.6842
-1.5714	-3.0203
-3.0360	-2.4470
-0.1221	-1.8903
-1.0593	-0.8713
-1.2127	-2.2900
-2.8759	-0.7384
-1.6801	-1.5246
-2.5583	-0.8259
-2.3114	-1.8731
-2.5700	-2.6568
-3.0257	-3.4814
-2.9087	-1.8445
-2.2099	-1.1814
-3.6989	-2.2926
-1.3924	-2.5408
-2.1178	-2.3086
-1.3008	-3.0966
-1.7304	-2.4930
-1.5057	-2.1807
-3.4831	-1.9542

Data generation
Two centers

```
>> x=
[randn(20,2)+ones(20,1)*y(1,:);randn(20,2)+ones(2
0,1)*y(2,:)];
>> plot(x(:,1),x(:,2),'o')
```

Forward data generation of two centers

$$y_1, y_2 \in R^d$$

$$S_1 = \{x_i = y_1 + n_i\}$$

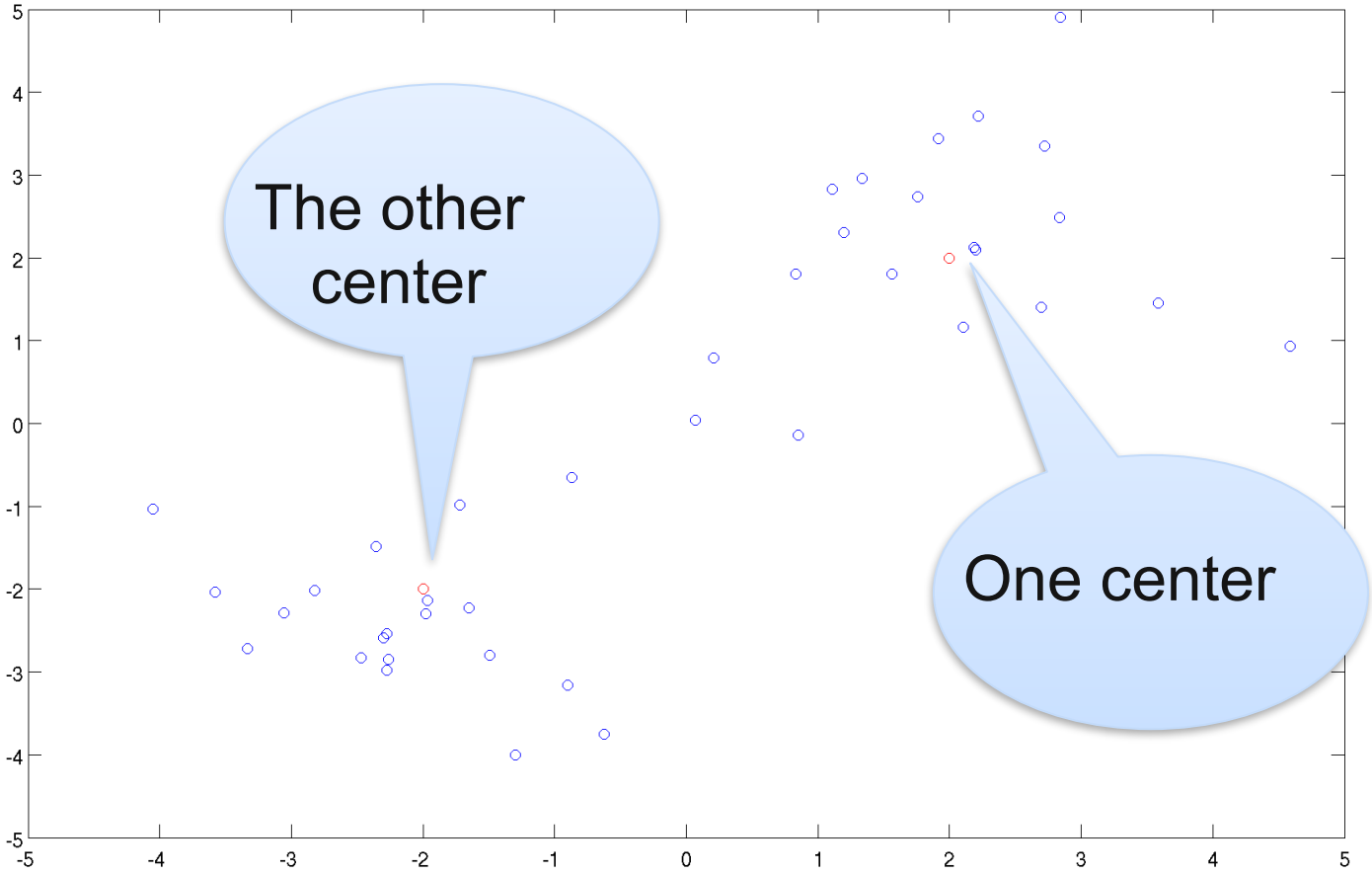
$$S_2 = \{x_i = y_2 + n_i\}$$

$$S = S_1 \cup S_2$$

Data in S1 centered at y1
Data in S2 centered at y2

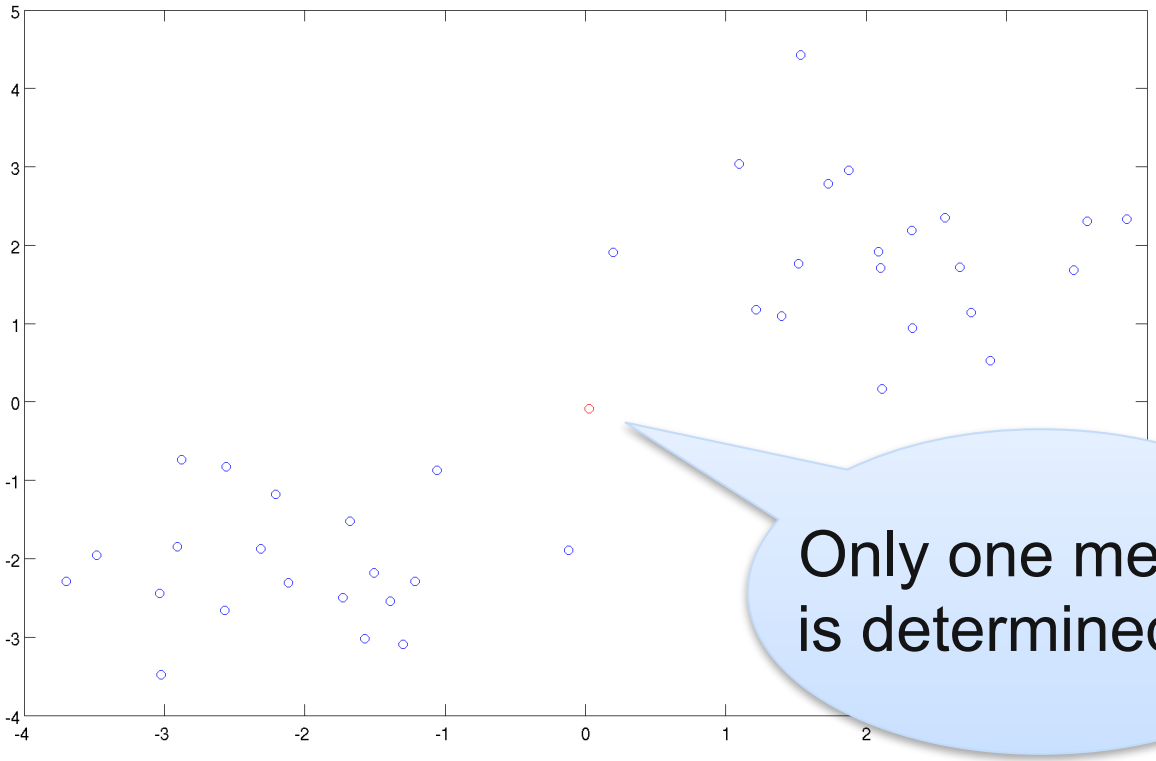
S contains mixtures of samples from two PDFs

Mixture data from two pdfs of different means
Blue circles
Red centers



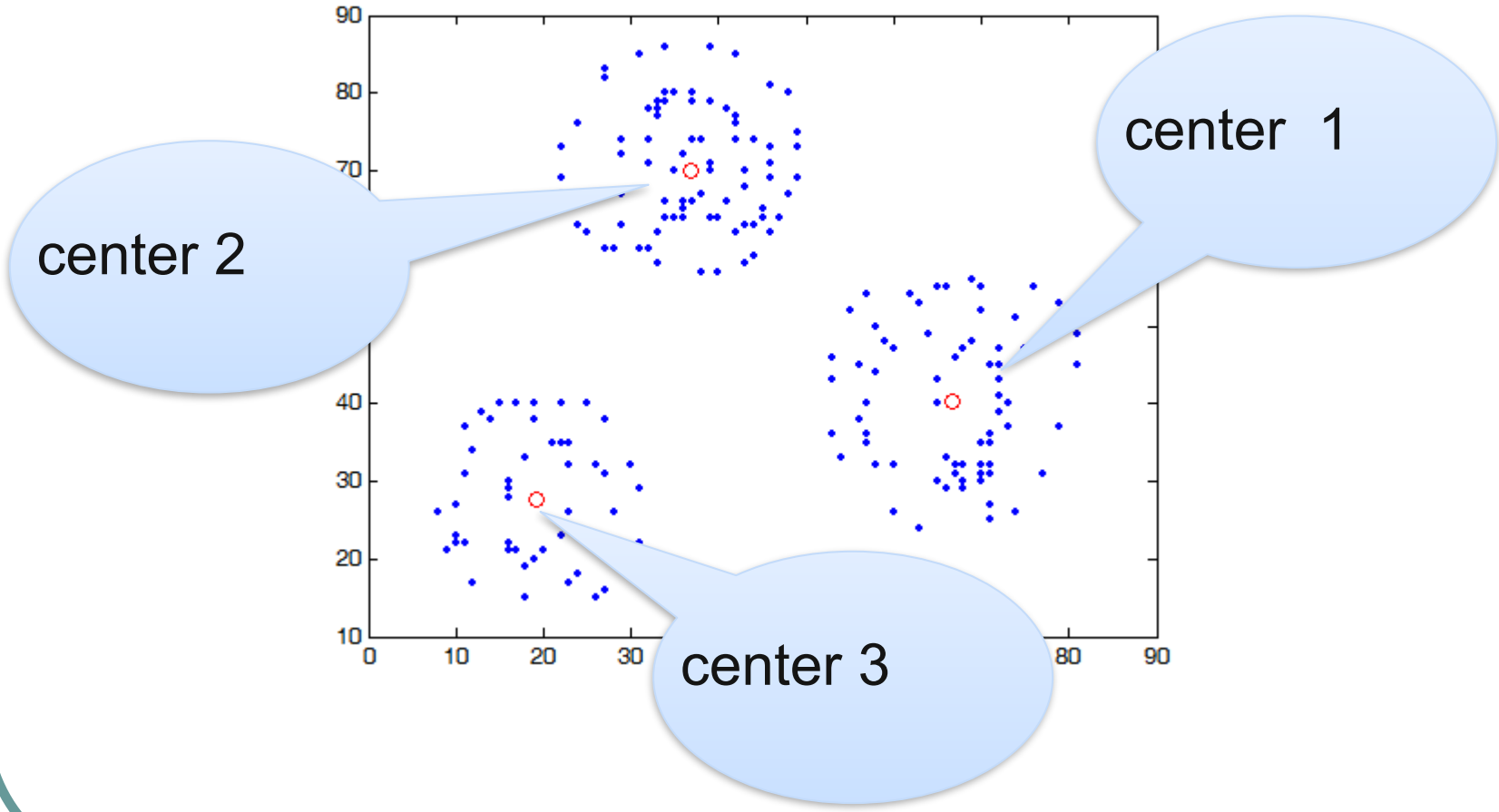
```
>> centers=mean(x);
>> hold on;plot(centers(:,1),centers(:,2),'ro')
```

Instruction 'mean' is not able to find two centers
How to retrieve two centers?



Only one mean is determined

Three centers



Data generation: Mixtures of K PDFs

S_k contains data centered at y_k

S contains data with K centers

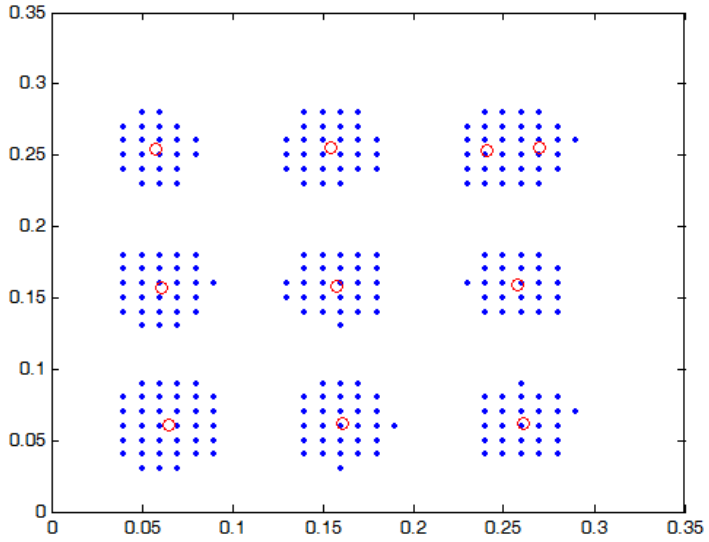
$$S_k = \{x_i = y_k + n_i\}$$

$$k = 1, \ldots, K$$

$$S = S_1 \cup S_2 \cdots \cup S_K$$

K centers

In general, there are K centers for characterizing mixture data
How to find centers for given K?



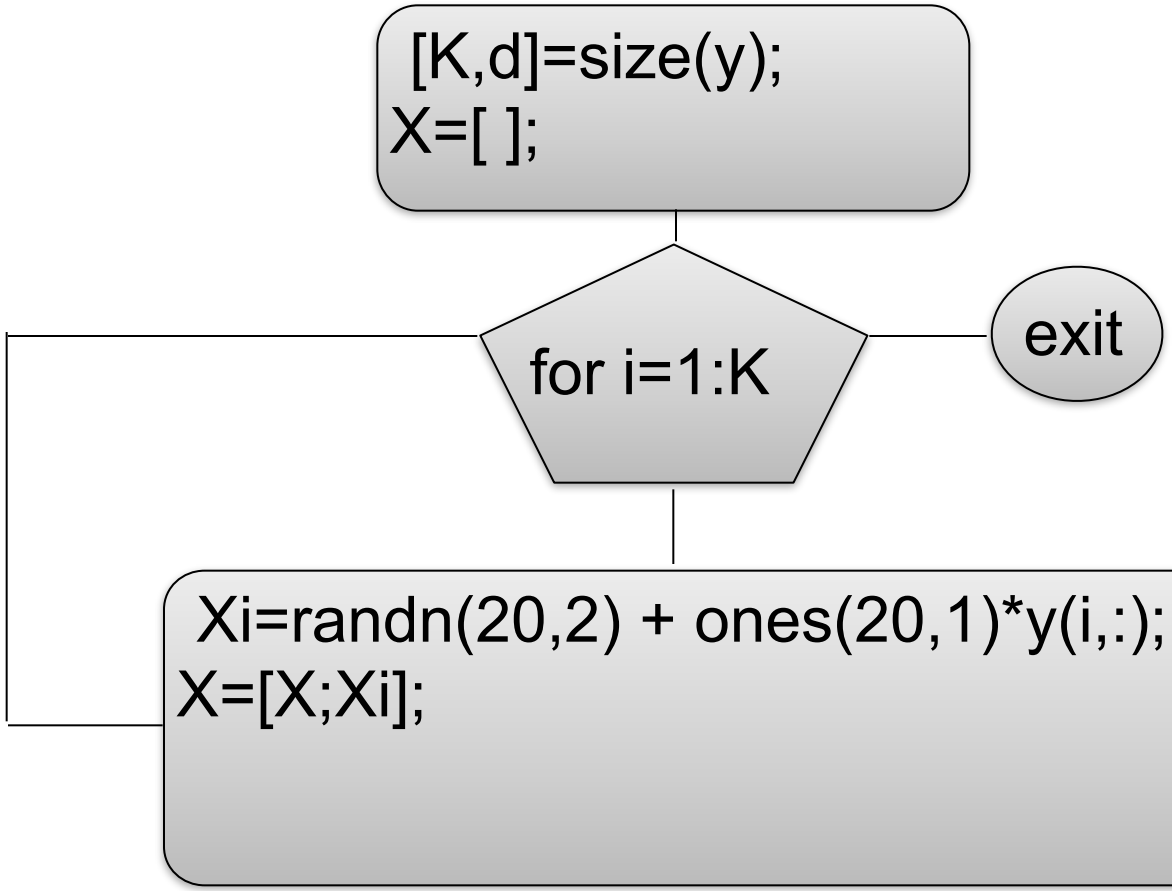
Computational tasks

- Data generation: use K centers to generate mixture data.
- Calculations of cross distances between K centers and N points
- Determine exclusive memberships of N points to K centers
- An iterative approach to refine K centers
- Define criteria of qualifying K centers

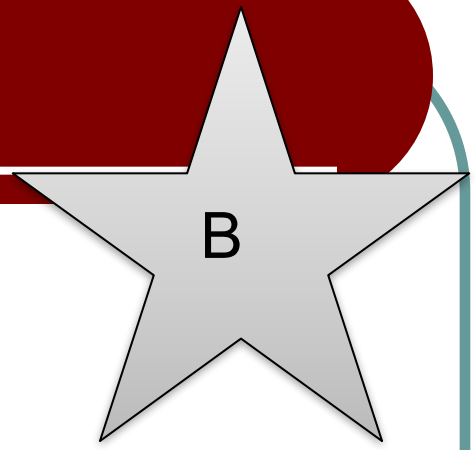
Data structure

- y denotes a Kxd matrix. y(k,:) denotes the mean of the kth pdf
- X denotes a Nxd matrix. X(i,:) denotes the position of the ith point
- Y denotes a Kxd matrix. Y(k,:) denotes the position of the kth center

Data generation



Initialization




```
mean_x = mean(X);
Y=randn(K,d)*0.01+ones(K,1)*mean_x;
```

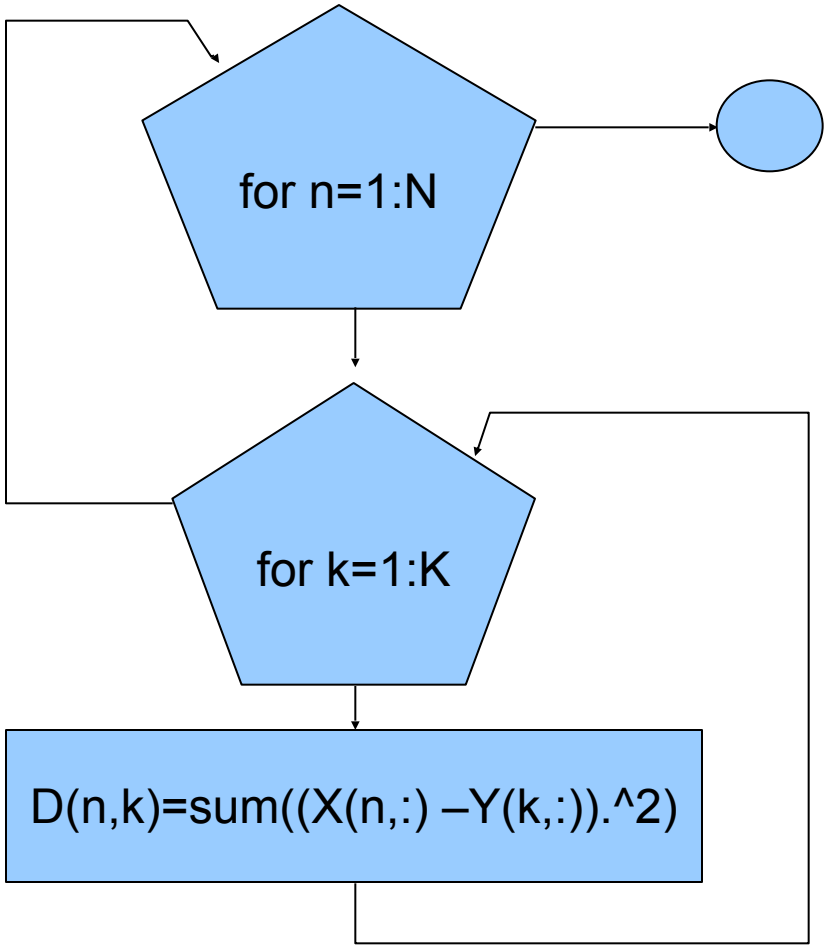
Cross distances

- How to find distances between given points and centers?

Calculation of Cross distances

- Given N points X: Nx2
- K centers Y: Kx2
- D: NxK
- D(i,j) denotes the distance between X(i,:) and Y(j,:)
- Given X and Y, find D

Nested loops for cross distances



Matlab codes for nested for-looping

```
for i=1:N
    for j=1:M
        dd=X(i,:)-Y(j,:);
        D(i,j)=sqrt(sum(dd.^2));
    end
end
```

- Straightforward implementation
- Nested looping
  - A loop within a loop
  - KN calculations of the distance between a point and a center
- Time consuming for large K,N and d

Vector codes

- Vector codes are loop-free
- Vector codes are efficient for large N,K in computation

$$D_{ij} = (\mathbf{x}_i - \mathbf{y}_j)(\mathbf{x}_i^T - \mathbf{y}_j^T)$$

$$= \mathbf{x}_i\mathbf{x}_i^T - 2\mathbf{x}_i\mathbf{y}_j^T + \mathbf{y}_j\mathbf{y}_j^T$$

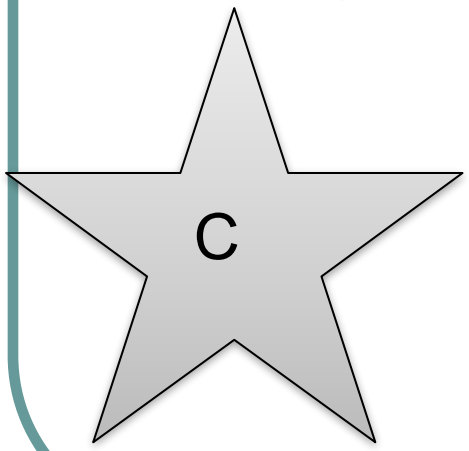
$$= A_{ij} - 2B_{ij} + C_{ij}$$

D : cross distances between N points and M centers
Matrix D is decomposed to matrices A, B and C
A : elements in a row are identical
B : multiplication of matrix X and transpose of matrix Y
C : elements in a column are identical

$$D_{ij} = (\mathbf{x}_i - \mathbf{y}_j)(\mathbf{x}_i^T - \mathbf{y}_j^T)$$

$$= \mathbf{x}_i\mathbf{x}_i^T - 2\mathbf{x}_i\mathbf{y}_j^T + \mathbf{y}_j\mathbf{y}_j^T$$

$$= A_{ij} - 2B_{ij} + C_{ij}$$




```
K=size(Y,1);N=size(X,1);
A=sum(X.^2,2)*ones(1,K);
C=ones(N,1)*sum(Y.^2,2)';
B=X*Y';
D=sqrt(A-2*B+C);
```

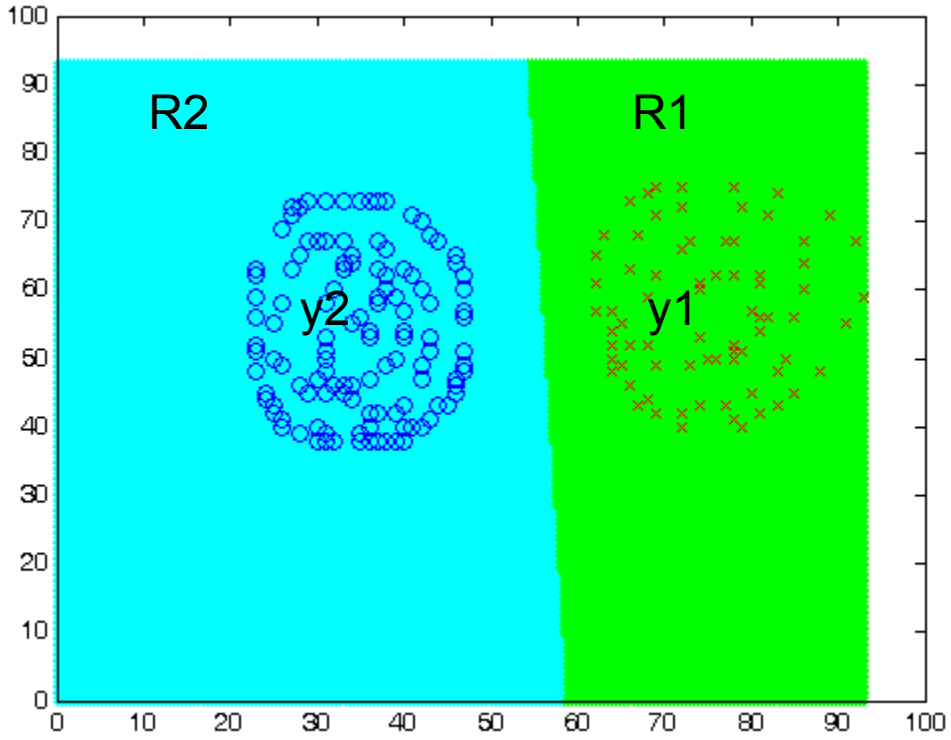
Vector codes of cross distances

demo_distance2.m

```
 7 - for i=1:N
 8 -     for j=1:M
 9 -         dd=X(i,:)-Y(j,:);
10 -         D(i,j)=sqrt(sum(dd.^2));
11 -     end
12 - end
13 - ss2=cputime;
14 - A=sum(X.^2,2)*ones(1,M);
15 - C=ones(N,1)*sum(Y.^2,2)';
16 - B=X*Y';
17 - DD=sqrt(A-2*B+C);
18 - sum(sum(abs(DD-D)))
```

Partition to two regions

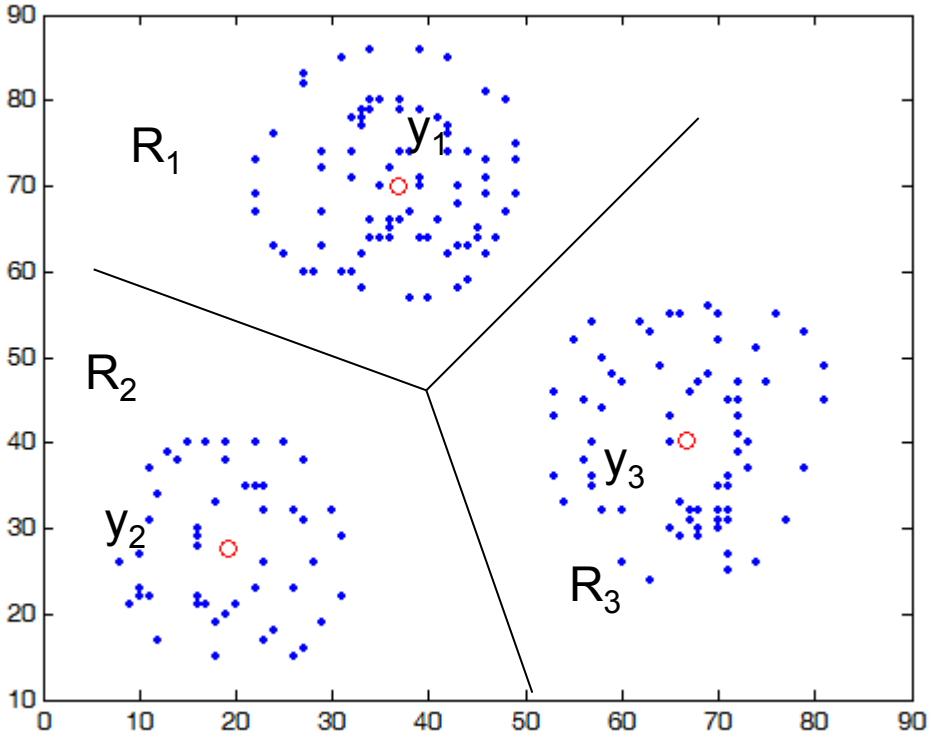
Each point has its **exclusive membership** to non-overlapping regions partitioned by two centers



Exclusive membership

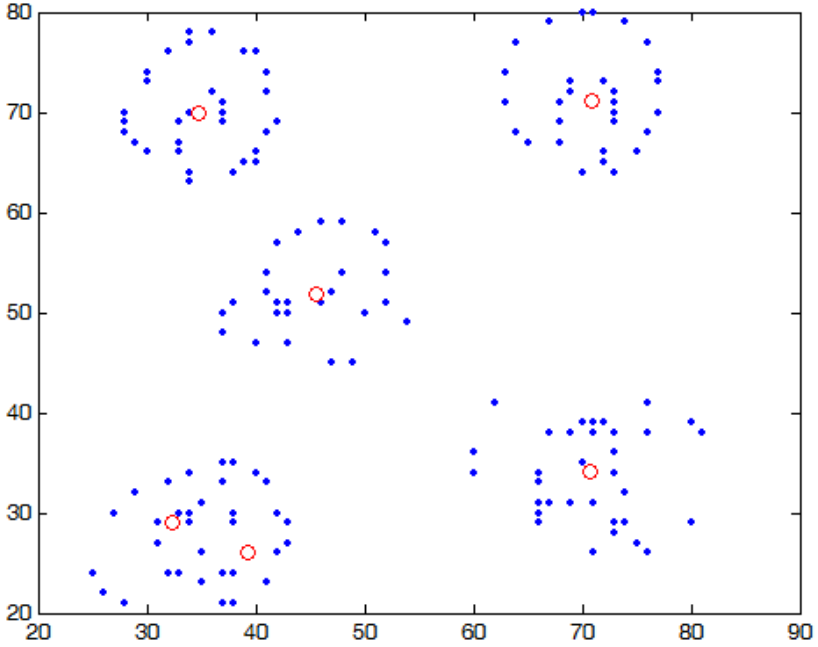
- y_1 and y_2 denote two centers
- R_1 and R_2 denote two regions partitioned by y_1 and y_2
- A point belongs to R_1 if it is closer to y_1
- A point belongs to R_2 if it is closer to y_2

Three clusters



K clusters

- Locating K centers
- Significant geometric features of points in R^d



Exclusive membership

- $y_1, y_2, \ldots, y_K$ denote K distinct centers
- $R_1, R_2, \ldots, R_K$ denote K regions partitioned by $y_1, y_2, \ldots, y_K$
- A point belongs to R_i if it is closest to y_i among K centers

- Let D denote cross distances between N given points and K centers
- Find points nearest to the jth center

Point x(i,:) belongs the jth cluster if

$$\mathrm{D}(i,j) = \min_{k} D(i,k)$$

Exclusive memberships

- Given D, exclusive memberships v of N points can be determined by

$$D = \begin{bmatrix} 3 & 1 & 0 \\ 4 & 2 & 1 \end{bmatrix}$$

[dd v] = min(D')

v : [3 3]

dd : [0 1]

v(i) denotes the index of the minimal entry in the ith row of matrix D.
v(i) denotes an exclusive membership of point i to K clusters.

```
[dd  v]=min(D');
```

Exclusive memberships and new centers

- Collect points belonging the kth cluster

```
ind=find(v == k);
```

- These points defines a new center

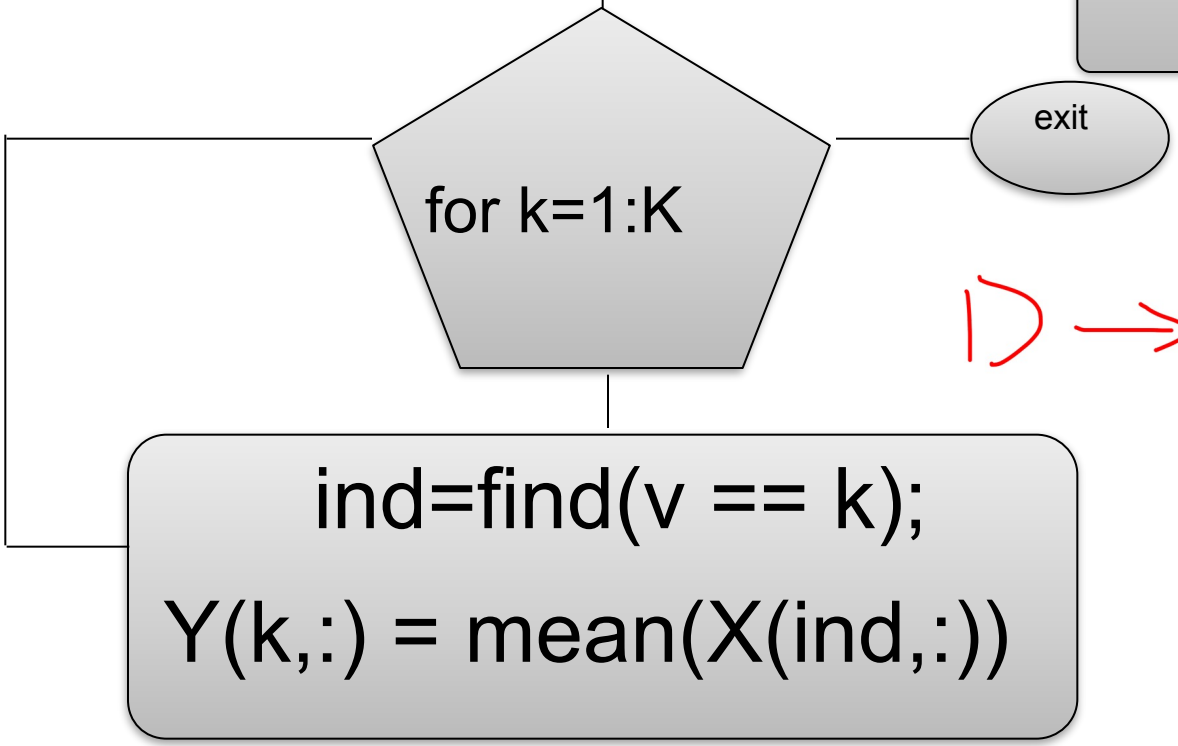
```
Y(j,:) = mean(X(ind,:))
```

Exclusive memberships, Criterion, Refining K centers


```
[dd  v]=min(D');
mean(dd)
```

$$E = \frac{1}{N} \sum_{i} \min_{j} \|x_i - y_j\|$$

mean(dd)

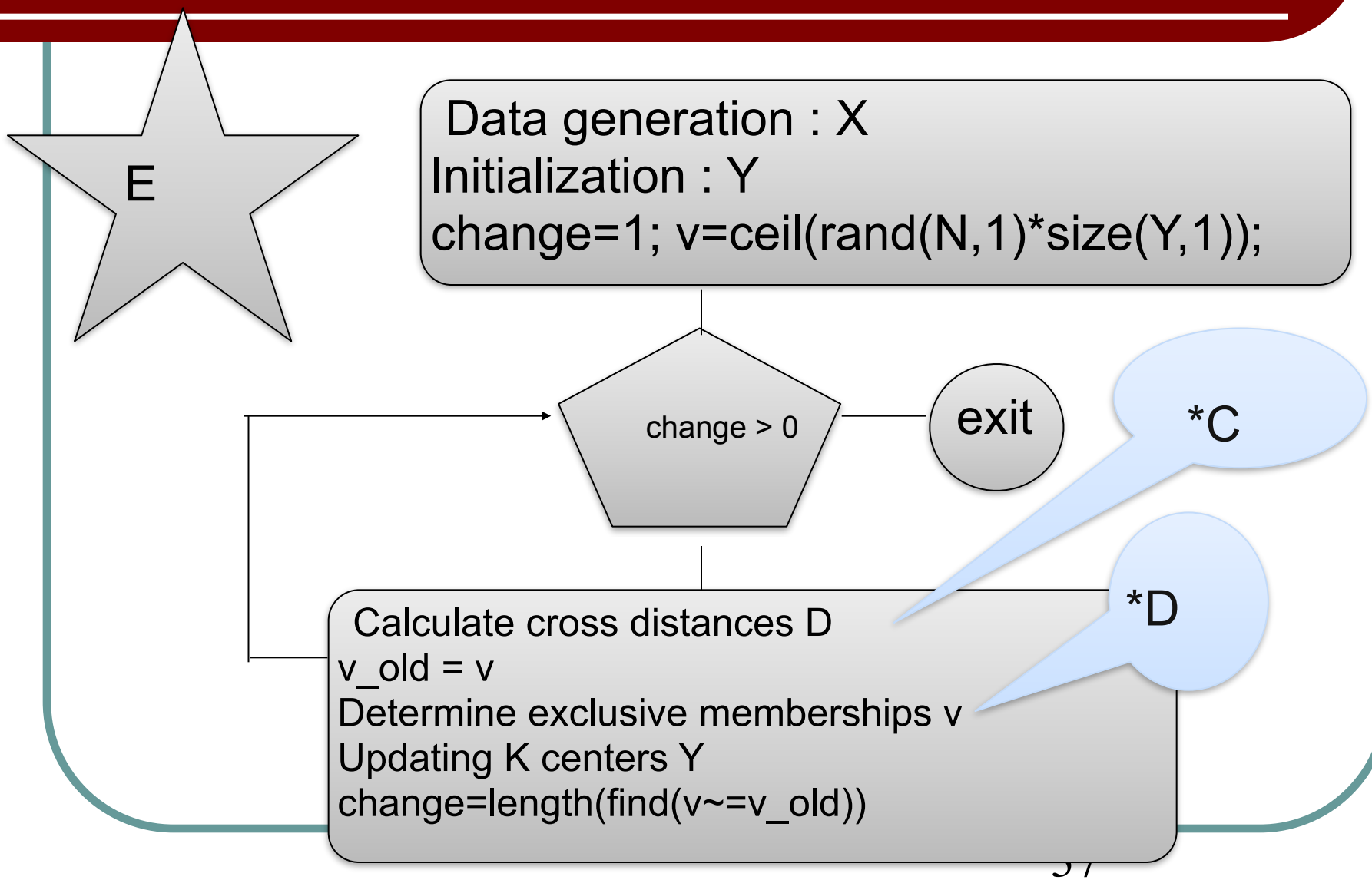


D → V dd → Y E

An iterative approach for K-means

- A popular heuristic approach for clustering
- An iterative approach
  - Step A : cross distance D
  - Step B : exclusive memberships v
  - Step C : updating centers Y

An iterative approach for K-means

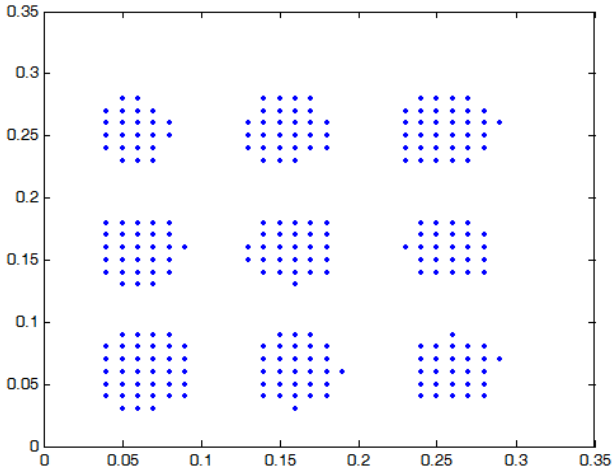


Data and MATLAB codes

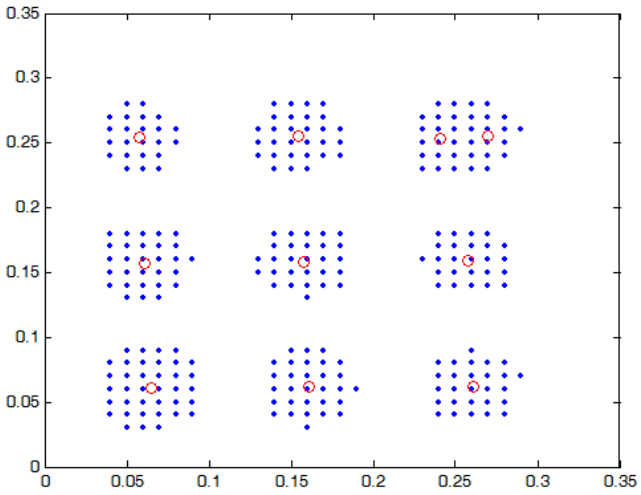
data_9.zip
demo_kmeans.m

```
load data_9.mat
plot(X(:,1),X(:,2),'.');
[cidx, Y] = kmeans(X,10);
hold on;
plot(Y(:,1),Y(:,2),'ro');
```

Data Clustering



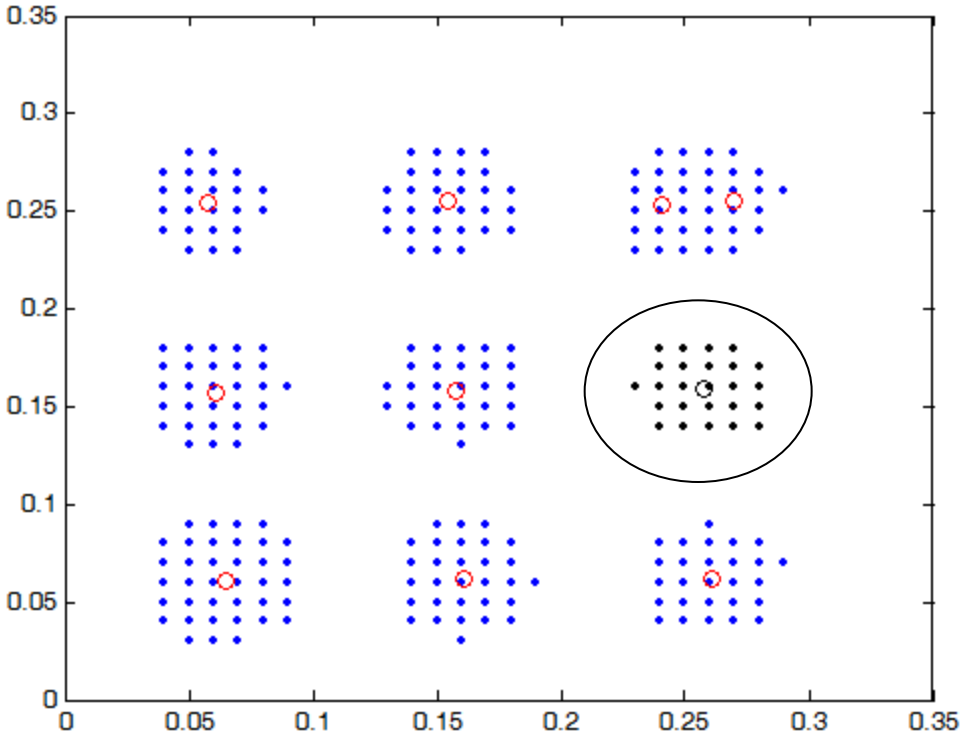
[cidx, Y] = kmeans(X,10);



- Partition given data to K clusters
- The K-means algorithm aims to find means (centers) of K clusters

Memberships

Black points belong to the cluster centered at black circle



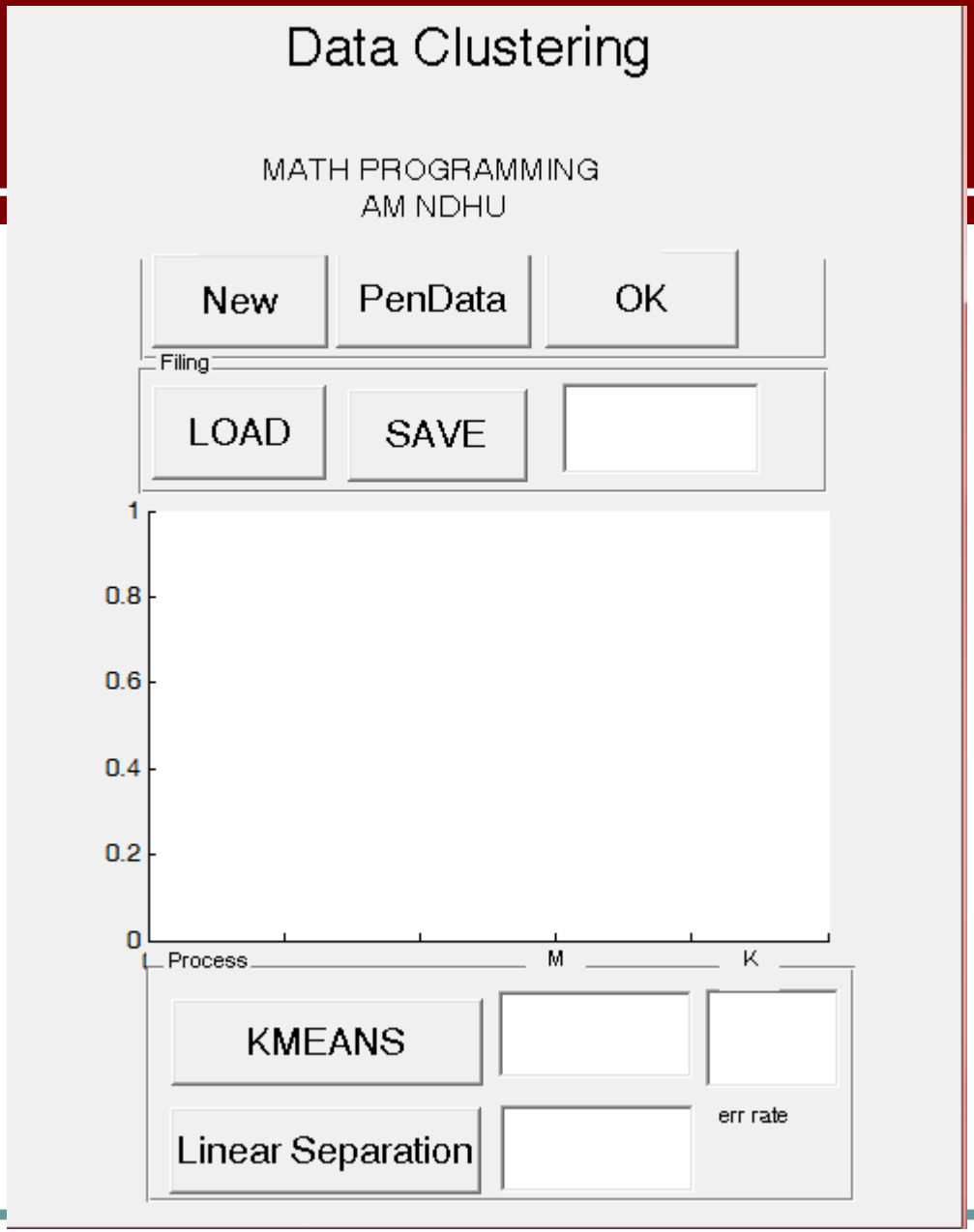
A math tool for data clustering

ClusteringTest.fig

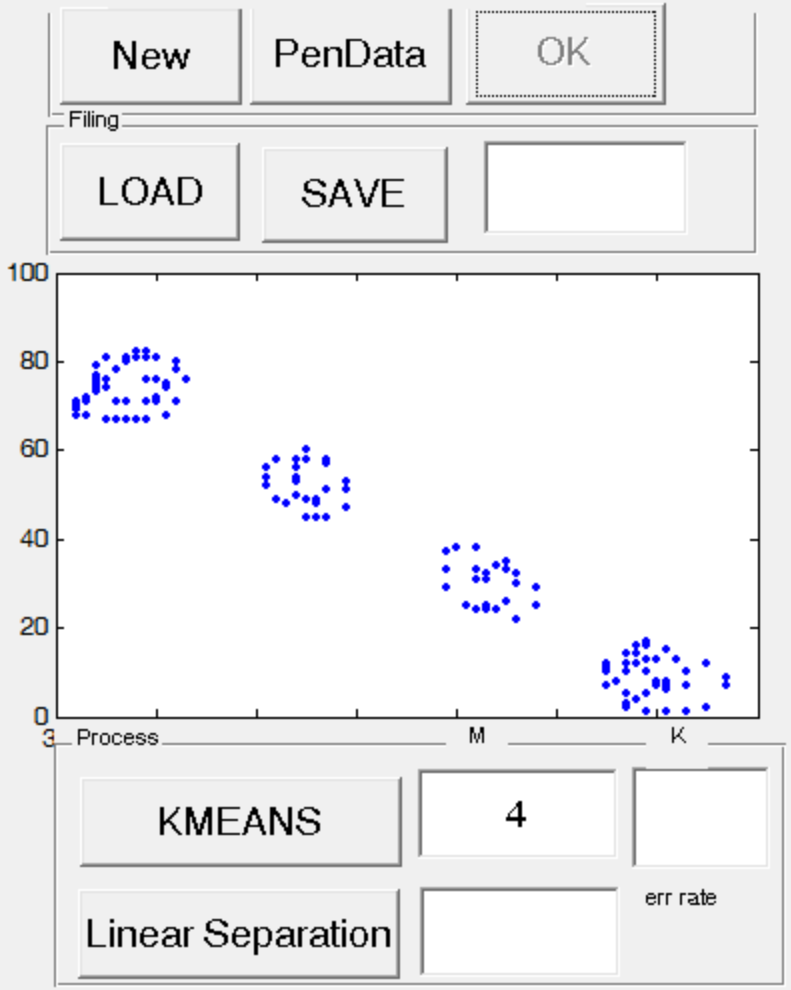
ClusteringTest.m

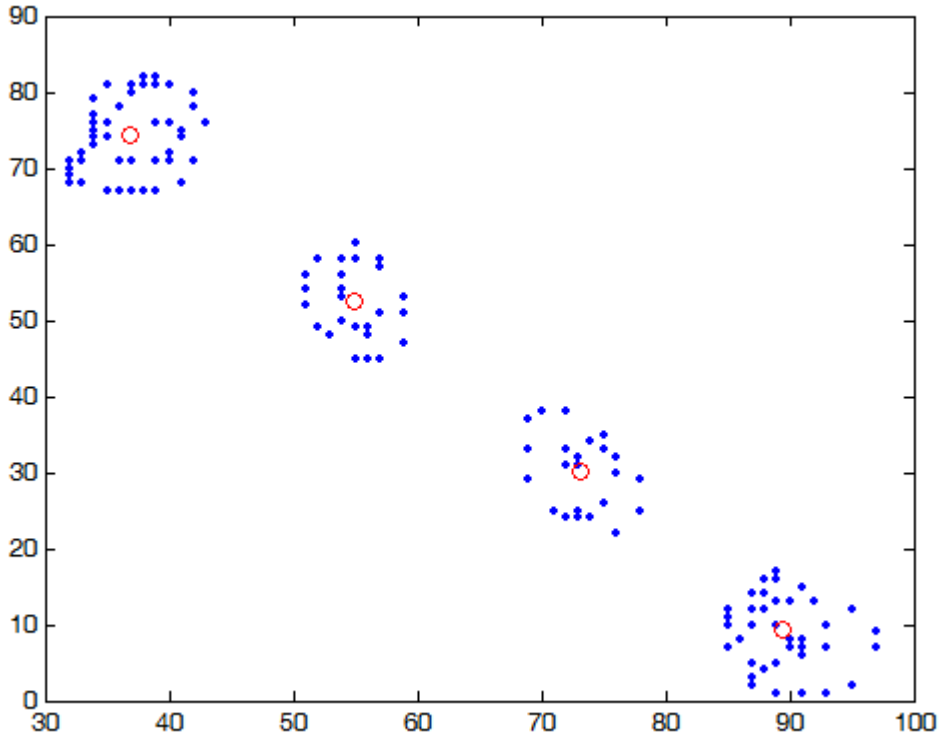
Steps for data clustering:

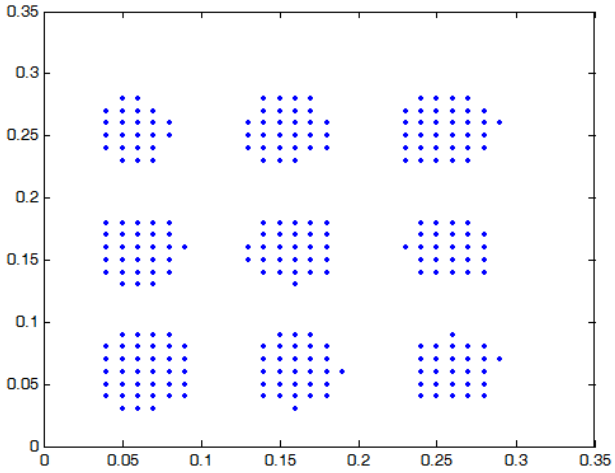
New
PenData
Enter M
KMEANS



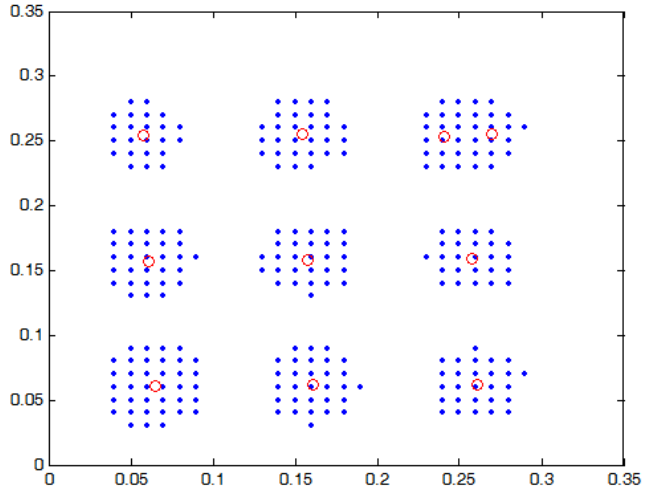
Data Clustering




90
80
70
60
50
40
30
20
10
0
30
40
50
60
70
80
90
100



[cidx, Y] = kmeans(X,10);

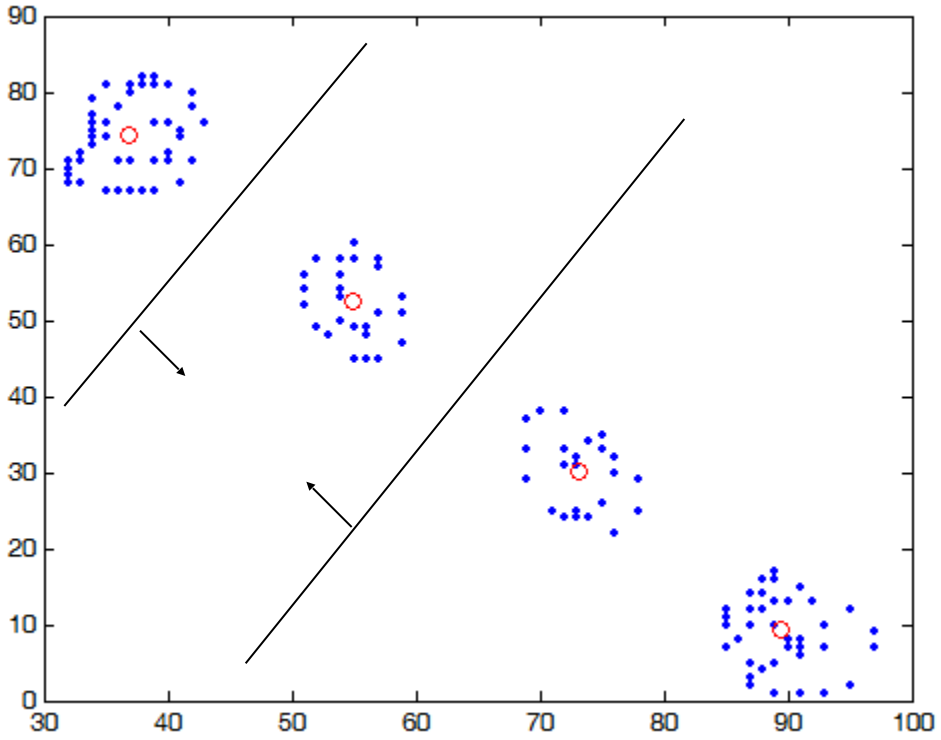



```
M=size(Y,1);N=size(X,1);
A=sum(X.^2,2)*ones(1,M);
C=ones(N,1)*sum(Y.^2,2)';
B=X*Y';
D=sqrt(A-2*B+C);
```

Cross Distances

Membership

- X(:,i) belongs a cluster



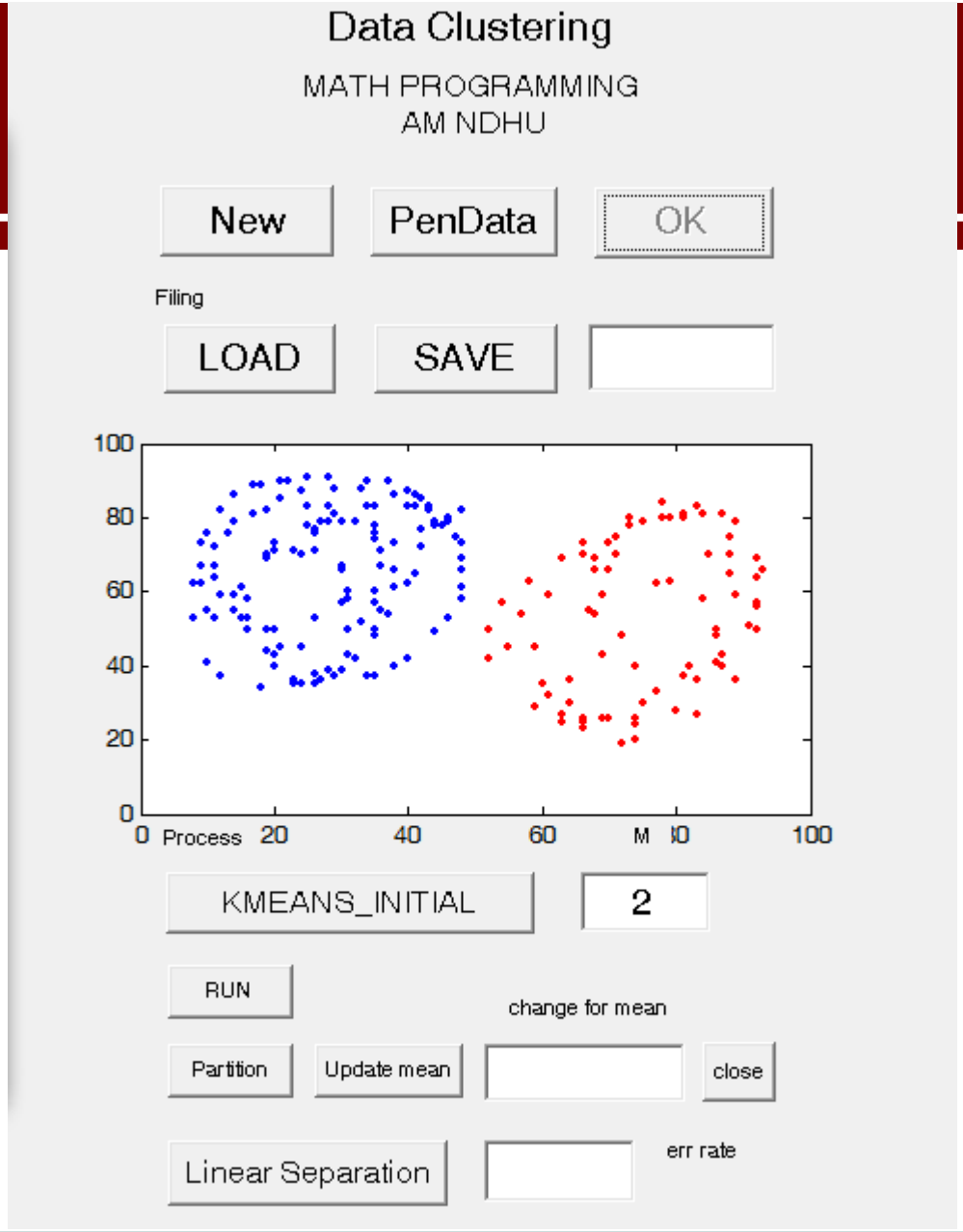
my_kmeans

my_kmeans.m

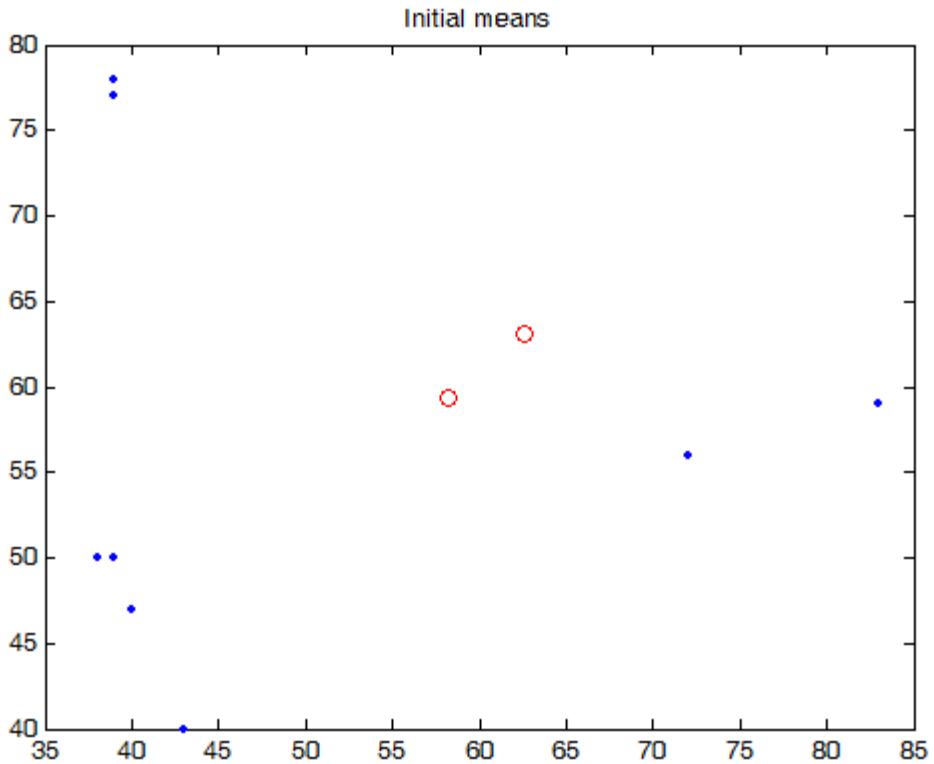
```
load data_9.mat
>> plot(X(:,1),X(:,2),'.');
>> Y=my_kmeans(X,10);
>> hold on
>> plot(Y(:,1),Y(:,2),'or');
```

ClusteringTest2.fig
ClusteringTest2.m

A version that is able to show stepwise execution of partitioning and updating of the kmeans algorithm


Data Clustering
MATH PROGRAMMING
AM NDHU
New
PenData
OK
Filing
LOAD
SAVE
100
80
60
40
20
0
0
Process
20
40
60
100
KMEANS_INITIAL
2
RUN
change for mean
Partition
Update mean
close
Linear Separation
err rate

Initialization



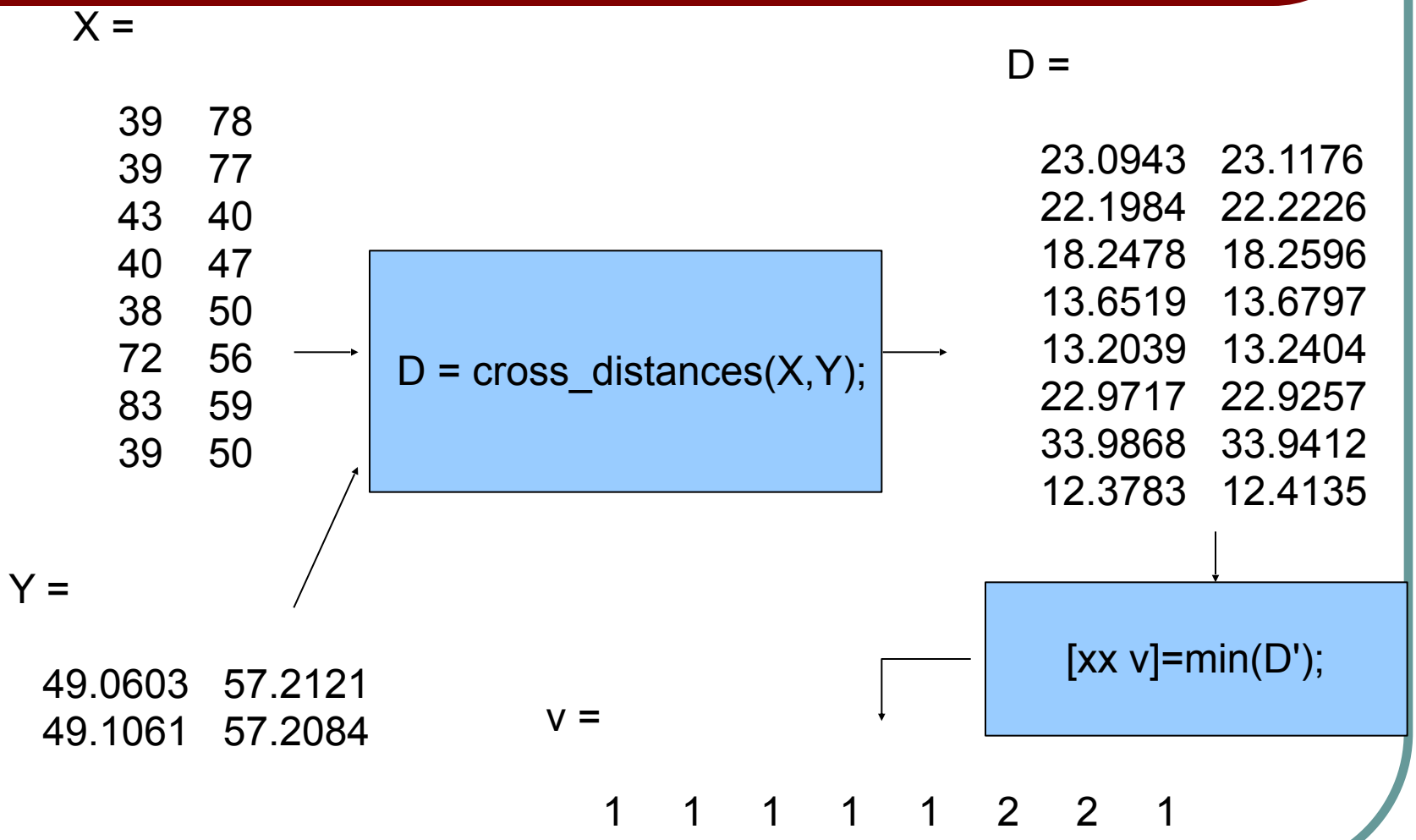
X =

39	78
39	77
43	40
40	47
38	50
72	56
83	59
39	50

Y =

49.0603	57.2121
49.1061	57.2084

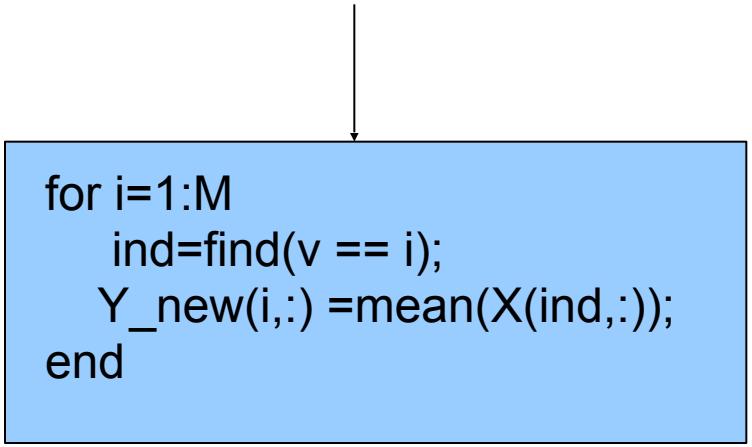
Cross distance



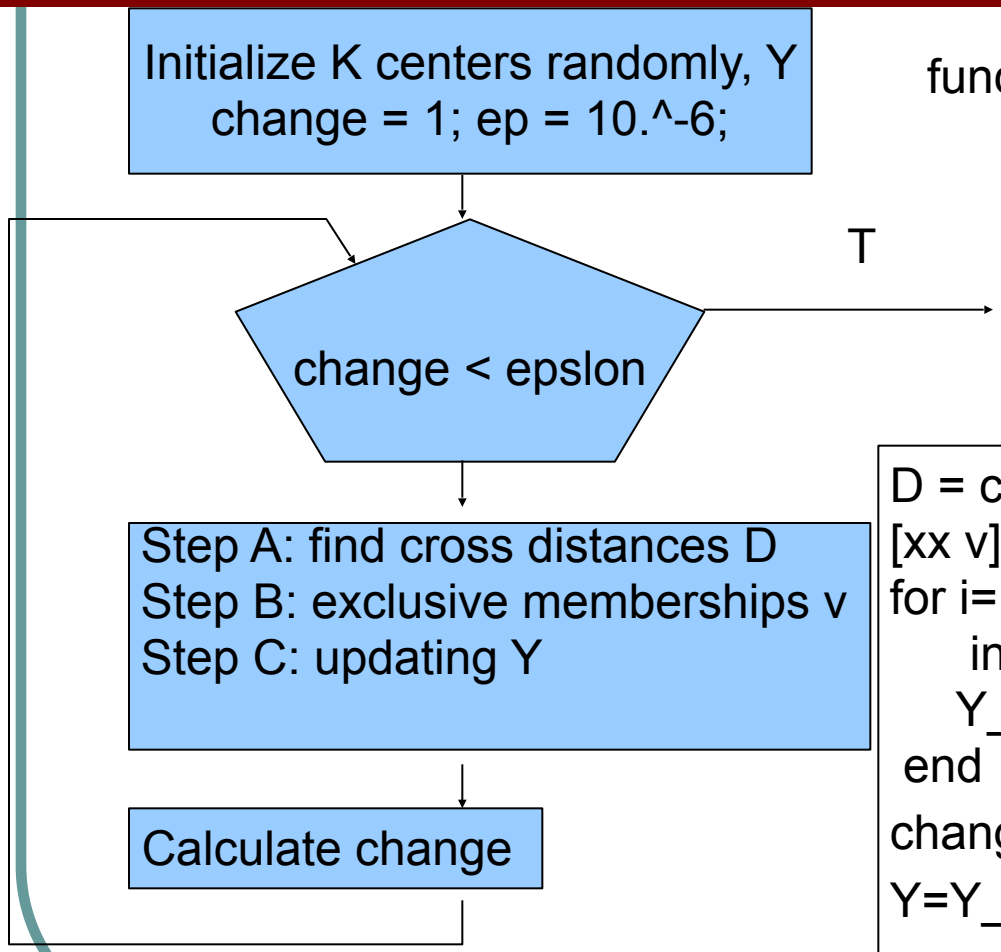
Partition & Update

ind =

1 1 1 1 1 2 2 1



```
for i=1:M
    ind=find(v == i);
   Y_new(i,:) =mean(X(ind,:));
end
```




```
function Y=my_kmeans(X,M)
```

```
D = cross_distances(X,Y);
[xx v]=min(D');
for i=1:M
      ind=find(v == i);
     Y_new(i,:) =mean(X(ind,:));
 end
change = mean(mean(abs(Y-Y_new)));
Y=Y_new;
```

k-means clustering - Wikipedia

Description

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, the *k*-means clustering aims to partition the *n* observations into *k* sets ($k < n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

History

The term "*k*-means" was first used by James MacQueen in 1967,[1] though the idea goes back to Hugo Steinhaus in 1956.[2] The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published until 1982.[3]

Standard Algorithm

Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).

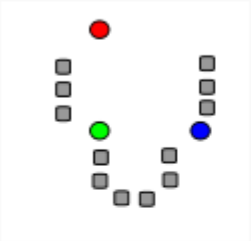
$$S_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \right\| \text{ for all } i^* = 1, \ldots, k \right\}$$

Update step: Calculate the new means to be the centroid of the observations in the cluster.

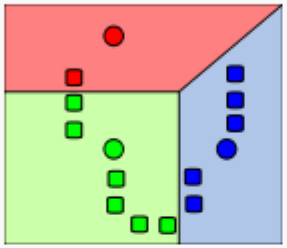
$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

The algorithm is deemed to have converged when the assignments no longer change.

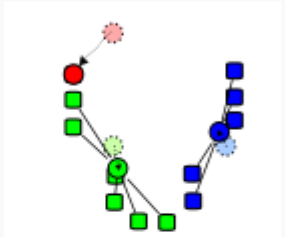
Demonstration of the standard algorithm



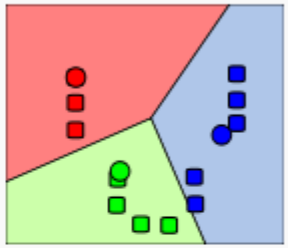
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



3) The centroid of each of the *k* clusters becomes the new means.



4) Steps 2 and 3 are repeated until convergence has been reached.

Advanced topics based on K-means

- Classification
- Function approximation
- Density estimation